Updated facet response

Purpose

The current CMR hierarchical facet API contains design decisions that promote a tight coupling between CMR clients and CMR data. In particular, clients must know information about which fields can be faceted, and which hierarchical facets can only be queried via the JSON query API. Because of this coupling, changes to the data returned by the CMR facet API tend to cause clients to break or degrade. This page proposes a protocol to decouple clients from facet data, simultaneously allowing flexibility for the CMR about which data to return while making the CMR response itself easier for clients to process. Further, this flexibility allows the CMR to provide more intelligent facet responses in the future, enabling facets to be tailored to search criteria.

There are two key uses for facets: support for collection discovery and support for controlled vocabularies. The former aids collection search for clients such as GCMD, Earthdata Search, and the NSIDC Search application, while the latter would aid a metadata authoring client like the MMT. The protocols described here are designed for the collection search use case only, and it is likely that a client like the MMT would require a different method for discovering facets.

Approach

The protocol we build uses well-established principles of hypermedia REST APIs, in particular it seeks to:

1. Define a media type (response format) and processing rules to express facet responses
2. Provide extensibility while maintaining forward-compatibility for clients so new types of facets can be expressed without breaking existing clients
3. Allow clients to process facet responses without knowing specifics of the data they contain, and, in fact, explicitly discourage clients from making assumptions about data contents
4. Provide straightforward, convenient parsing and processing rules for clients, allowing them to easily display, apply, and un-apply facet terms

To accomplish these goals, we will provide a simple tree-like response structure. Each tree node contains type information, allowing clients to process data differently, and allowing the CMR to extend the format in the future without breaking clients. The structure itself contains enough information to present facets without any *a priori* knowledge of which facets are returned, their order, or their format. Finally, the format is simple to parse. We provide an example client that can display facet information in both EDSC- and GCMD-like views.

Requesting the Updated Format

Clients may request the updated format using the parameter "include_facets=v2". Clients specifying "include_facets=true" or "include_facets=v1" will continue to receive the old, deprecated facet response until it is removed from the CMR. Clients requesting "include_facets=v2" do not need to specify "hierarchical_facets=true" because the response format is implicitly hierarchical.

Response Format

When querying the CMR JSON API and requesting facets, the response will contain a field, "facets" containing the root node of the aforementioned tree structure. Every node in the tree structure contains the following minimal structure:

```
var treeNode = {
  "title": "Example",         // A human-readable representation of the
node
  "type": "group|filter|..."  // The type of node represented
};
```

Currently, the filter response type defines two node types: "group" and "filter". More may be added in the future, and clients must be built to ignore unknown node types.

Group Nodes

Group nodes act as containers for similar data. Depending on context, this data may, for instance, be all facet parameters (the root facet node), top-level facets, or children of existing facets. Group nodes further have a

```
var groupNode = { // Inherits treeNode fields
  "applied": true,            // true if the filter represented by this
node (see Filter Nodes) or any of its descendants has been applied to the
current query
  "has_children": true,       // true if the tree node has child nodes,
false otherwise
  "children": [               // List of child nodes, provided at the
discretion of the CMR (see below)
  ]
}
```

Children

In order to avoid sending unnecessary information, the CMR may opt to not return children for group nodes that have facets, returning only the first few levels of the facet tree. It does, however, provide three guarantees which allow clients to appropriately display incomplete information and query the full tree as necessary:

1. The initial facet tree will contain, at a minimum, the root node, its most relevant (see below) children (the top-level filter categories), and the first ***facet_min_depth*** levels of the most relevant filter nodes below those children.
2. If a faceted filter is applied, clients are guaranteed that it will appear in the tree, and the CMR will return the next ***facet_min_depth*** levels of the most relevant nodes below that facet (if applicable)
3. If a node has children, but all are filtered by the current query, the "has_children" flag will be false. This prevents clients from inappropriately showing that a node has children, only to reveal that it doesn't after re-querying the CMR.

Where ***facet_min_depth*** is 2 by default, but may be overridden by clients as a query parameter, e.g. "...&facet_min_depth=4". Setting ***facet_min_depth*** to 0 instructs the CMR to return the whole tree, despite the potentially very large response size.

Relevance

By default, clients should assume that the CMR may limit facet results to only include the most relevant child nodes in facet responses. For instance, if there are hundreds of science keywords at a particular depth, the CMR may choose to only return those that have a substantial number of results. Clients may override this by explicitly setting the ***facet_limit_children*** flag to false. When filtering children, the CMR makes no guarantees about the specific quantities or values of facets returned, only that applied filters attempt to surface the choices that typical users are most likely to find beneficial.

Filter Nodes

Filter nodes are group nodes that supply a single query condition indicated by a string (the node title). They further have a field, "applied," which indicates if the query value has already been applied to the current query.

To allow removal of filters, applied filters are guaranteed to appear in the facet tree, even if the current query has no collection results.

```
var filterNode = { // Inherits groupNode fields
  "count": 1234,
// Count of results matching the filter
  "links": {
// Provided only for clients using the query parameter API
    "apply": "https://example.com/search/collections?instrument[]=MODIS",
// A URL containing the filter node applied to the current query. Returned
only if the node is not applied.
    "remove": "https://example.com/search/collections"
// A URL containing the filter node removed from the current query.
Returned only if the node is applied.
  },
  "queries" {
// Provided only for clients using the JSON query API
    "apply": {"instrument": "MODIS"},
// A query object containing the filter node applied to the current query.
Returned only if the node is not applied.
    "remove": {}
// A query object containing the filter node removed from the current
query. Returned only if the node is applied.
  }
}
```

Note that while four potential queries are listed between "links" and "queries", in practice only one would be returned based on the request's API and whether the node is currently applied.

Full Example

The following example is a sample response for a query using the query parameters API which has the "instruments=ASTER" filter applied as well as a page size of 10 applied by the client. The example is hand-constructed for example purposes only. Real-world queries would typically be more complex, counts would be different, and the facet tree would be much larger.

```
{
  "facets": {
    "title": "Browse Collections",
    "type": "group",
    "applied": true, // NOTE: true because the tree does have a descendent
node that has been applied
    "has_children": true,
    "children": [
      {
        "title": "Instruments",
        "type": "group",
        "applied": true,
        "has_children": true,
        "children": [
          {
            "title": "MODIS",
            "type": "filter",
            "applied": false,
```

```
            "count": 200,
            "links": { "apply":
"https://example.com/search/collections?page_size=10&instruments[]=ASTER&i
nstruments[]=MODIS" },
            "has_children": false
          },
          {
            "title": "ASTER",
            "type": "filter",
            "applied": true,
            "count": 2000,
            "links": { "remove":
"https://example.com/search/collections?page_size=10" }, // NOTE: Differing
response for an applied filter
            "has_children": false
          }
        ]
      },
      {
        "title": "Keywords",
        "type": "group",
        "applied": false,
        "has_children": true,
        "children": [
          {
            "title": "EARTH SCIENCE",
            "type": "filter",
            "applied": false,
            "count": 1500,
            "links": { "apply":
"https://example.com/search/collections?page_size=10&instruments[]=ASTER&s
cience_keywords[0][category_keyword]=EARTH%20SCIENCE" },
            "has_children": true,
            "children": [ // NOTE: This is an example of how the CMR may
handle children at different levels of a hierarchy.
              {           //       For usability reasons, this should only
be done if absolutely necessary, preferring to just collapse the hierarchy
when possible
                "title": "Topics",
                "type": "group",
                "applied": false,
                "has_children": true,
                "children": [
                  {
                    "title": "OCEANS",
                    "type": "filter",
                    "applied": false,
                    "count": 500,
                    "links": { "apply":
"https://example.com/search/collections?page_size=10&instruments[]=ASTER&s
cience_keywords[0][category_keyword]=EARTH%20SCIENCE&science_keywords[0][t
opic]=OCEANS" },
                    "has_children": true // NOTE: The node has children,
```

```
but the CMR has opted not to return them
                  }
                ]
              },
              {
                "title": "Variables",
                "type": "group",
                "applied": false,
                "has_children": true,
                "children": [
                  {
                    "title": "WOLVES",
                    "type": "filter",
                    "applied": false,
                    "count": 2,
                    "links": { "apply":
"https://example.com/search/collections?page_size=10&instruments[]=ASTER&s
cience_keywords[0][detailed_variable]=WOLVES" },
                    "has_children": false
                  }
                ]
              }
            ]
          }
        ]
      }
    ]
```

```
    }
};
```

Error rendering macro 'pageapproval' : null